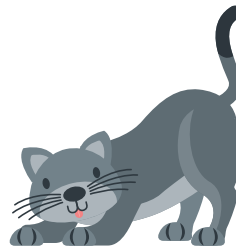
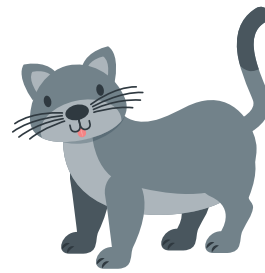
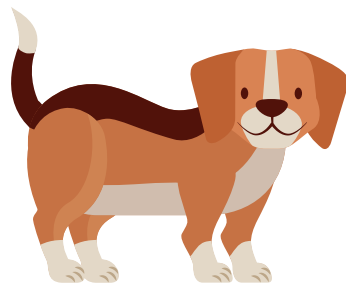


MIAUHEAVEN PROFILING



*María Jesús Cadenas Sánchez
Sara Donaire Guillen
Alfonso González Fernández
Francisco Javier González Lugo
Rafael Salas Castizo
Javier Solís García*

PROFILING

HU.07

En las pruebas de rendimiento de la historia de usuario 7 se puede observar que con pocos usuarios ya se consigue llegar a las condiciones óptimas de nuestra web. Por lo que nos hace pensar que algo debe estar ocurriendo para que se esté usando tan pocos usuarios para llegar a las condiciones óptimas del sistema debe de haber un cuello de botella en el código en el que se implementa esta historia de usuario. Gracias al uso de Glowroot puede observar que al convertir un Owner en animalShelter, al llamar al Owner este llamaba a sus Pets y estas Pets traían consigo todas las visitas que en este caso no son necesarias llamarlas.

Para esta función se hace la siguiente consulta de las Pets del Owner:

```
SELECT pet0_.id AS id1_6_0_,
       pet0_.name AS name2_6_0_,
       pet0_.birth_date AS birth_da3_6_0_,
       pet0_.genre AS genre4_6_0_,
       pet0_.owner_id AS owner_id5_6_0_,
       pet0_.type_id AS type_id6_6_0_,
       owner1_.id AS id1_5_1_,
       owner1_.first_name AS first_na2_5_1_,
       owner1_.last_name AS last_na3_5_1_,
       owner1_.address AS address4_5_1_,
       owner1_.city AS city5_5_1_,
       owner1_.telephone AS telephon6_5_1_,
       owner1_.username AS username7_5_1_,
       user2_.username AS username1_12_2_,
       user2_.enabled AS enabled2_12_2_,
       user2_.password AS password3_12_2_,
       pettype3_.id AS id1_11_3_,
       pettype3_.name AS name2_11_3_,
       visits4_.pet_id AS pet_id4_15_4_,
       visits4_.id AS id1_15_4_,
       visits4_.id AS id1_15_5_,
       visits4_.visit_date AS visit_da2_15_5_,
       visits4_.description AS descript3_15_5_,
       visits4_.pet_id AS pet_id4_15_5_
FROM   pets pet0_
       LEFT OUTER JOIN owners owner1_ ON pet0_.owner_id = owner1_.id
       LEFT OUTER JOIN users user2_ ON owner1_.username = user2_.username
       LEFT OUTER JOIN types pettype3_ ON pet0_.type_id = pettype3_.id
       LEFT OUTER JOIN visits visits4_ ON pet0_.id = visits4_.pet_id
WHERE  pet0_.id = ?
```

Para evitar que Pet traiga consigo todos sus datos como las visitas que tiene asociadas hacemos lo siguiente en la entidad de Pet:

```
@OneToMany(cascade = CascadeType.ALL, mappedBy = "pet", fetch = FetchType.LAZY)
private Set<Visit> visits;
```

De esta forma, al añadir el LAZY, el Pet si no es necesario no traerá consigo todos los datos relacionados con Visitas. Ahora la consulta que se haría sería la siguiente:

```
SELECT pet0_.id AS id1_6_0_,
       pet0_.name AS name2_6_0_,
       pet0_.birth_date AS birth_da3_6_0_,
       pet0_.genre AS genre4_6_0_,
       pet0_.owner_id AS owner_id5_6_0_,
       pet0_.type_id AS type_id6_6_0_,
       owner1_.id AS id1_5_1_,
       owner1_.first_name AS first_na2_5_1_,
       owner1_.last_name AS last_nam3_5_1_,
       owner1_.address AS address4_5_1_,
       owner1_.city AS city5_5_1_,
       owner1_.telephone AS telephon6_5_1_,
       owner1_.username AS username7_5_1_,
       user2_.username AS username1_12_2_,
       user2_.enabled AS enabled2_12_2_,
       user2_.password AS password3_12_2_,
       pettype3_.id AS id1_11_3_,
       pettype3_.name AS name2_11_3_
FROM   pets pet0_
       LEFT OUTER JOIN owners owner1_ ON pet0_.owner_id = owner1_.id
       LEFT OUTER JOIN users user2_ ON owner1_.username = user2_.username
       LEFT OUTER JOIN types pettype3_ ON pet0_.type_id = pettype3_.id
WHERE  pet0_.id = ?
```

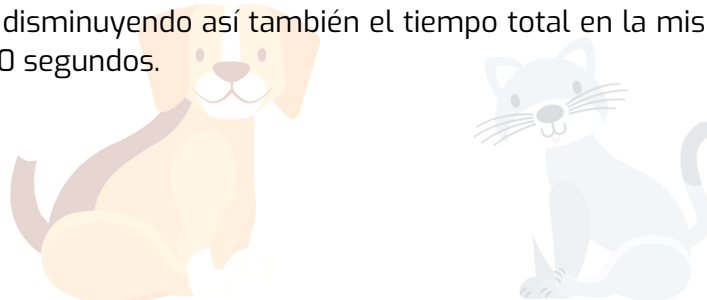
Para tener una buena media de los tiempos he hecho uso de gatling para poner a varios usuarios a realizar estas acciones. Sin los cambios realizados tenemos estos resultados:

```
select pet0_.id as id1_6_0_, pet0_.name as name2_6_0_, pet0_.birth_date as 1300,9 20,000 0.065 1,0
birth_da...
```

El tiempo medio es de 0.065 ms, haciendo un total de 13000 ms. En cambio, tras realizar los cambios comentados previamente, obtenemos los siguientes resultados:

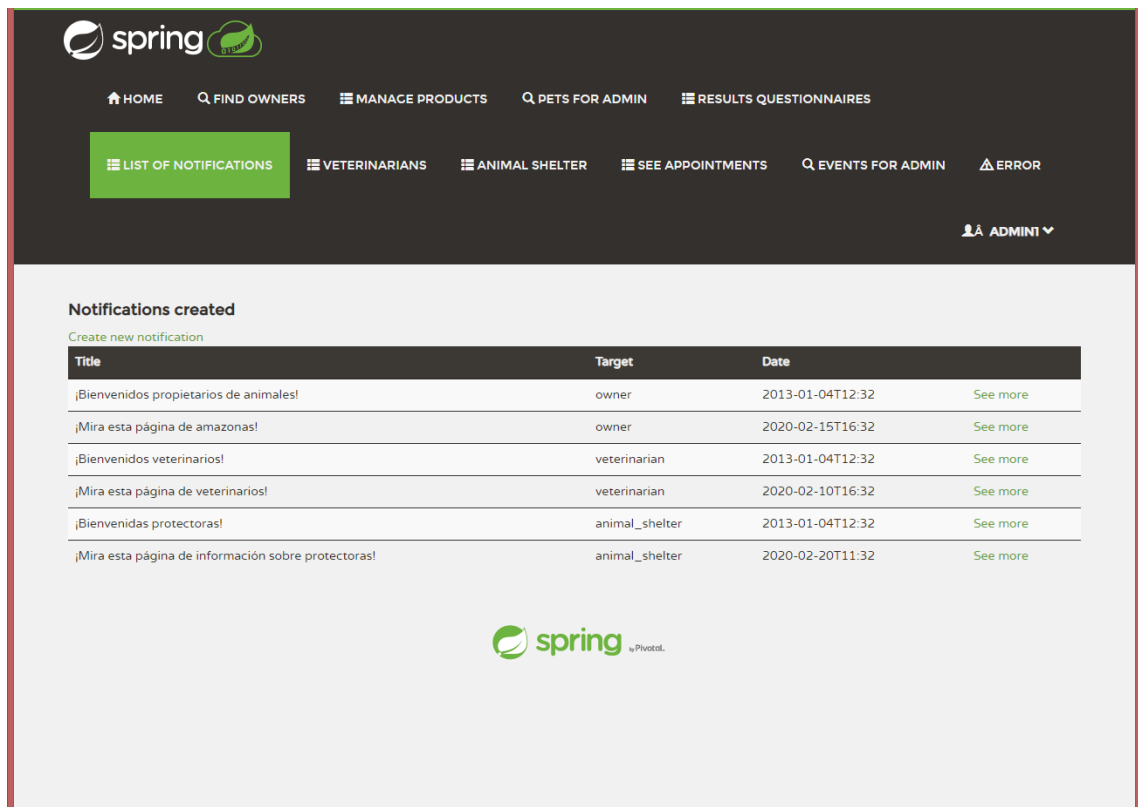
```
select pet0_.id as id1_6_0_, pet0_.name as name2_6_0_, pet0_.birth_date as 1020,5 20,000 0.051 1,0
birth_da...
```

Aquí podemos apreciar que hemos disminuido el tiempo medio en 0.01s (de 0.065 a 0.051), disminuyendo así también el tiempo total en la misma cantidad, de 13 segundos a 10 segundos.



En las pruebas de rendimiento de esta historia de usuario, pudimos observar que sólo con 750 usuarios, un número demasiado bajo, se llegaba a las condiciones óptima de nuestra aplicación.

El cuello de botella viene determinado por la cantidad de notificaciones a mostrar, ya que el administrador es quien crea estas notificaciones, las lista todas y las elimina.



spring

HOME FIND OWNERS MANAGE PRODUCTS PETS FOR ADMIN RESULTS QUESTIONNAIRES

LIST OF NOTIFICATIONS VETERINARIANS ANIMAL SHELTER SEE APPOINTMENTS EVENTS FOR ADMIN ERROR

ADMIN

Notifications created

Create new notification

Title	Target	Date	
¡Bienvenidos propietarios de animales!	owner	2013-01-04T12:32	See more
¡Mira esta página de amazonas!	owner	2020-02-15T16:32	See more
¡Bienvenidos veterinarios!	veterinarian	2013-01-04T12:32	See more
¡Mira esta página de veterinarios!	veterinarian	2020-02-10T16:32	See more
¡Bienvenidas protectoras!	animal_shelter	2013-01-04T12:32	See more
¡Mira esta página de información sobre protectoras!	animal_shelter	2020-02-20T11:32	See more

spring Pivotal

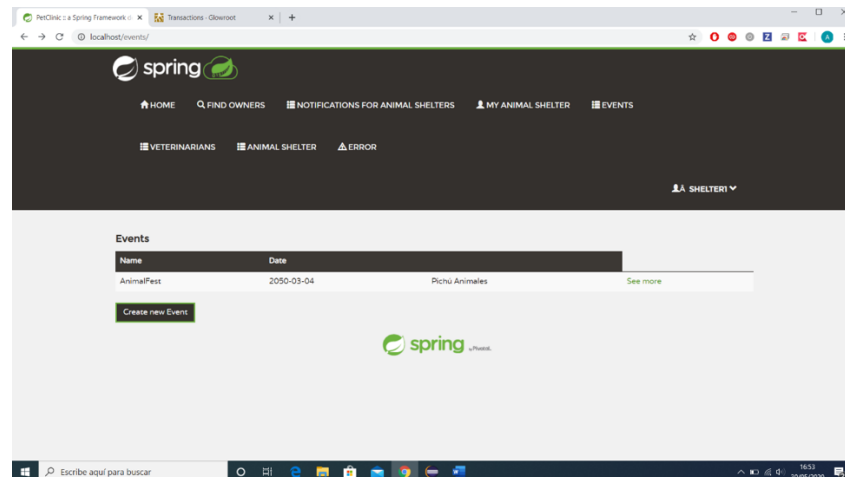
Como podemos observar, es una consulta simple por lo que en principio no debería dar problemas.

```
SELECT notificati@.id AS id1_4_,
       notificati@.date AS date2_4_,
       notificati@.message AS message3_4_,
       notificati@.target AS target4_4_,
       notificati@.title AS title5_4_,
       notificati@.url AS url6_4_
FROM notification notificati@_
(show unformatted)
```

Como las notificaciones solo son creadas y borradas por los administradores, vamos a añadir que solo se borre la caché cuando una nueva notificación sea creada. Con esto evitaremos que se estén actualizando continuamente.

HU.23

En las pruebas de rendimiento de la historia de usuario 23 vimos como necesitábamos un número muy bajo de usuarios para llegar a las condiciones óptimas de nuestra aplicación. Viendo eso, me dio que pensar que tenía que haber un cuello de botella bastante notable en algún punto. Mediante el uso de Glowroot vi que al acceder a la página en la que se muestran los eventos, para ser únicamente un listado con el nombre del evento, el nombre del Animal Shelter y la fecha. La vista de la que hablo es la siguiente:



Para esta vista, la consulta que se está realizando es la siguiente:

```
SELECT event0_.id AS id1_3_,
       event0_.name AS name2_3_,
       event0_.animalshelter_id AS animalsh5_3_,
       event0_.date AS date3_3_,
       event0_.description AS descript4_3_
FROM   events event0_
```

(show unformatted)

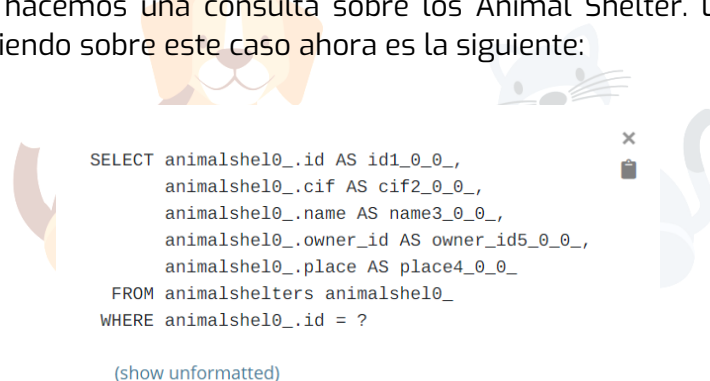
En esta consulta se estarían devolviendo los eventos con sus datos, de momento nada raro. El problema viene en la consulta que se hace para obtener la información del animal shelter, que se muestra a continuación:

```
SELECT animalshel0_.id AS id1_0_0_,
       animalshel0_.cif AS cif2_0_0_,
       animalshel0_.name AS name3_0_0_,
       animalshel0_.owner_id AS owner_id5_0_0_,
       animalshel0_.place AS place4_0_0_,
       owner1_.id AS id1_5_1_,
       owner1_.first_name AS first_na2_5_1_,
       owner1_.last_name AS last_nam3_5_1_,
       owner1_.address AS address4_5_1_,
       owner1_.city AS city5_5_1_,
       owner1_.telephone AS telephon6_5_1_,
       owner1_.username AS username7_5_1_,
       user2_.username AS username1_12_2_,
       user2_.enabled AS enabled2_12_2_,
       user2_.password AS password3_12_2_
FROM   animalshelters animalshel0_
LEFT OUTER JOIN owners owner1_ ON animalshel0_.owner_id = owner1_.id
LEFT OUTER JOIN users user2_ ON owner1_.username = user2_.username
WHERE  animalshel0_.id = ?
```

Animal Shelter es una entidad diseñada por nosotros a modo de rol. Esta protectora de animales es un usuario más del sistema (es decir, un owner con todas las características de estos), que a parte tiene ciertos privilegios, como poder aceptar las peticiones de adopción a sus mascotas, o, como en este caso, crear eventos. Para la vista que queremos mostrar, los datos que necesitamos de la protectora es el nombre, y nos estamos trayendo todos los datos de Animal Shelter, y todos los datos de owner, que incluyen también los de user. Para evitar esto, vamos a añadir una notación a la relación de Animal Shelter con Owner:

```
Animalshelter.java
34  * @author Ken Krebs
35  */
36  @Entity
37  @Data
38  @EqualsAndHashCode(callSuper = false)
39  @Table(name = "animalshelters")
40  public class Animalshelter extends BaseEntity {
41
42      @Column(name = "name")
43      @NotEmpty
44      private String name;
45
46      @Column(name = "cif")
47      @NotEmpty
48      @Pattern(regexp = "\\d{8}[A-HJ-NP-TV-Z]")
49      private String cif;
50
51      @Column(name = "place")
52      @NotEmpty
53      private String place;
54
55      @OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
56      private Owner owner;
57
58  }
```

Al añadirle que el FetchType sea Lazy (ya que por defecto estaba como EAGER), le estamos diciendo que, si no es necesario, no se traiga la información referida a Owner cuando hacemos una consulta sobre los Animal Shelter. La consulta que estaríamos haciendo sobre este caso ahora es la siguiente:



```
SELECT animalshel0_.id AS id1_0_0_,
       animalshel0_.cif AS cif2_0_0_,
       animalshel0_.name AS name3_0_0_,
       animalshel0_.owner_id AS owner_id5_0_0_,
       animalshel0_.place AS place4_0_0_
FROM   animalshelters animalshel0_
WHERE  animalshel0_.id = ?
```

(show unformatted)

Como se ve en la captura, solo tiene la primera parte de la consulta anterior (la referida a Animal Shelter).

Para tener una buena media de los tiempos he hecho uso de gatling para poner a varios usuarios a realizar estas acciones. Sin los cambios realizados tenemos estos resultados:

```
select animalshel0_.id as id1_0_0_, animalshel0_.cif as cif2_0_0_,
animalshel0_.n...
```

3000,0	45,732	0.066	1,0
--------	--------	-------	-----

El tiempo medio es de 0.066 ms, haciendo un total de 3000 ms. En cambio, tras realizar los cambios comentados previamente, obtenemos los siguientes resultados:

```
select animalshel0_.id as id1_0_0_, animalshel0_.cif as cif2_0_0_,
animalshel0_.n...
```

2009,7	45,750	0.044	1,0
--------	--------	-------	-----

Aquí podemos apreciar que hemos disminuido el tiempo medio en 1/3 (de 0.066 a 0.044), disminuyendo así también el tiempo total en la misma cantidad, de 3 segundos a 2 segundos.

Conclusión

Tras realizar las pruebas de rendimiento sobre la historia 23, detecté que necesitaba un número inusualmente bajo de usuarios en comparación con las demás historias que realicé (que también eran sobre creación).

Mediante el uso de Glowroot pude ver los tiempos de acceso necesitados para cada página, así como las queries que se estaban solicitando. Ahí me di cuenta de que, para mostrar únicamente el nombre de la protectora de animales, se estaban solicitando también datos sobre las relaciones de esta clase con Owner, y de está con User.

Al conseguir que no se pidiese información, conseguíamos reducir el tiempo de las consultas en un 33%. Este dato, sobre 3 segundos totales de consultas como las capturas que he puesto arriba, solo supone un segundo de diferencia, que es el tiempo que empleamos en pestañear dos o tres veces; pero, por ejemplo, si se realizaran un número muy elevado de peticiones, cuyo tiempo total sumase 10 minutos, tras la refactorización solo requeriría de algo más de 6 minutos y medio, lo cual es un tiempo más significativo.